



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Mushtaq Bahadur
Appl. No.: 09/808,460
Filed: March 14, 2001
Title: DATA IMPORTER

Grp./A.U.: 2171
Examiner: To Be Assigned

Assistant Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Assistant Commissioner for Patents, Washington, DC 20231 on

August 29, 2001

(Date of Deposit)

Roger N. Chauza

(Name of Person Mailing Document)

(Signature)

(Date of Signature)

SUBMISSION OF PRIORITY DOCUMENT

Enclosed for filing in the above-referenced U.S. Patent Application are the following:

1. Certified copy of the priority document (Australian Pat. App. No. PQ 6307 filed on March 15, 2000).
2. Copy of the Assignment to iLaunch Pty Ltd, executed on March 12, 2001 and the recordation cover sheet.

SUBMISSION OF PRIORITY DOCUMENT

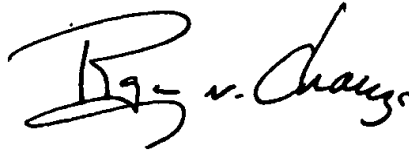
Atty. Dkt. No. ILAU-25,644

2171
#4

RECEIVED
SEP 06 2001
Technology Center 2100

3. Check in the amount of \$40.00 for the Assignment recordal fee. Please charge any additional fees or deficiencies in fees or credit any overpayment to Deposit Account No. 20-0780/ILAU-25,644 No. of HOWISON, CHAUZA, THOMA, HANDLEY & ARNOTT, L.L.P.

Respectfully submitted,
HOWISON, CHAUZA, THOMA, HANDLEY
& ARNOTT, L.L.P.
Attorneys for Applicant

A handwritten signature in black ink, appearing to read "Roger N. Chauza", written in a cursive style.

Roger N. Chauza
Registration No. 29,753

RNC:ljg

P.O. Box 741715
Dallas, Texas 75374-1715
Tele: (972) 479-0462
Fax: (972) 479-0464

August 29, 2001



Patent Office
Canberra

I, GAYE TURNER, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 6307 for a patent by I-LAUNCH PTY LTD filed on 15 March 2000.

RECEIVED
SEP 06 2001
Technology Center 2100

WITNESS my hand this
Nineteenth day of March 2001

GAYE TURNER
TEAM LEADER EXAMINATION
SUPPORT AND SALES

AUSTRALIA

Patents Act 1990

i-Launch Pty Ltd

PROVISIONAL SPECIFICATION

Invention Title:

Data Importer

The invention is described in the following statement:

Technical Field

This invention concerns the importation of data from external systems. In particular it concerns the importation of data from XML files.

5 Summary of the Invention

In a first aspect the invention, as currently envisaged, is a method for importing data from XML files, comprising the steps of:

Specifying an XML file to be imported.

Uploading the specified XML file.

10 Parsing the file into a series of values for graphically representing the structure of the data. For instance each node of an information (DOM) tree.

If necessary the values are corrected by a user inspecting the tree, into a format suitable to pass to the information tree

The tree may be viewed by a user.

15 The tree is saved by writing the data and metadata values to storage.

The storage may consist of four tables ie ww_form-temp (metadata). ww_form_item_temp (metadata). ww_files_temp (data) and ww_objects_temp (data).

20 The invention may be used to import and then view information from external systems. In a simple implementation an XML file may be imported without a DTD. Alternatively, in a more complex scenario the attributes of a corresponding DTD may be applied along with the presentation layer provided by XSL.

25 The information may be imported in batch or real-time mode from an external system such as Oracle Financials, SAP or Peoplesoft.

The imported information may be integrated with other systems without any code changes.

In another aspect, as currently envisaged, the invention is a computer system for importing data from XML files, comprising in data storage:

30 An Upload Servlet to upload a specified XML file.

A Parsing Servlet to decipher the file into a series of values for graphically representing the structure of the data.

A Saving Servlet to save the data and metadata values of the tree to storage.

35 In a further aspect, as currently envisaged, the invention is a computer program, comprising:

An Upload Servlet to upload a specified XML file.

A Parsing Servlet to decipher the file into a series of values for graphically representing the structure of the data.

5 A Saving Servlet to save the data and metadata values of the tree to storage.

Brief Description of the Drawings

An example of the invention will now be described with reference to the accompanying drawings, in which:

10 Fig. 1 is a flow chart showing the importation process.

Fig. 2 is a table showing the effect of parsing an XML file.

Fig. 3 is a table showing the structure of temporary storage tables.

Fig. 4 is a representation of forms that have been identified.

15 Fig. 5 is a representation of documents that could be produced.

Detailed Description of the Invention

Setting up of an importation interface involves installing server side utilities as well as a once-off client side modification. The modifications needed on the clients is simply a matter of installing the *Java Runtime Environment 1.2.2 (JRE)*, which includes appropriate plug-ins for both
20 Netscape Navigator 4.6+ (Navigator) and Internet Explorer 5+ (IE5). Once this is set up, all Java 1.2.2 applets will run in IE5 and Navigator.

Referring now to Fig. 1, the importation process 1 is started by a user calling a TrafficDirector Servlet 2 and specifying the XML file to be imported. This will
25 typically require typing in the host address, port number and database driver to be used. A username and password may be required to satisfy the login credentials for the external database. The TrafficDirector Servlet 2 then calls an Upload Servlet 3 and provides it with the appropriate parameters.

Once login to the external source has been achieved, then the
30 hostname and database name will appear, and a list of all the accessible tables will also be created, along with a list of all accessible columns from the selected table. This is the table where the data is to be retrieved from.

To limit the values which are available for selection, the user can create a criteria to determine which values will be available.

35 An XML document usually includes or contains a reference to a Document Type Definition (DTD). Essentially a DTD defines the grammar for

a class of documents, that is, it contains markup declarations that describe the documents logical structure and the constraints within this structure. An example of a DTD and a valid XML document written to this DTD is as follows. This example will be referred to throughout the remainder of this document:

Document Type Definition

```

    <!ELEMENT orderlist (order*)>
    <!ELEMENT order (datetime,notes,salesperson, customer, part*)>
    <!ATTLIST order id ID #REQUIRED>
10  <!ELEMENT datetime (#PCDATA)>
    <!ELEMENT notes (#PCDATA)>
    <!ELEMENT salesperson (name,department,phone)>
    <!ATTLIST salesperson id ID #REQUIRED>
    <!ELEMENT customer (name,address,phone)>
15  <!ATTLIST customer id ID #IMPLIED>
    <!ELEMENT part (name,quantity,price)>
    <!ATTLIST part id ID #REQUIRED>
    <!ELEMENT name (#PCDATA)>
    <!ELEMENT department (#PCDATA)>
20  <!ELEMENT phone (#PCDATA)>
    <!ELEMENT address (#PCDATA)>
    <!ELEMENT quantity (#PCDATA)>
    <!ELEMENT price (#PCDATA)>

```

Sample XML DOCUMENT

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE orderlist SYSTEM "orderlist.dtd">
<orderlist>
30  <order id="_5449431">
    <datetime>Feb 1 2000 5:37PM</datetime>
    <notes>We need to hurry this order through...</notes>
    <salesperson id="37">
        <name>Jill Smith</name>

```

```

    <department>Sales</department>
    <phone>90991234</phone>
  </salesperson>
  <customer id="909921">
5    <name>Bobs Plumbing</name>
    <address>1 George St, Sydney, 2000</address>
    <phone>90995678</phone>
  </customer>
  <part id="10987">
10    <name>Widget Flange</name>
    <quantity>100</quantity>
    <price>0.50</price>
  </part>
  <part id="10990">
15    <name>Widget Head Bolt</name>
    <quantity>100</quantity>
    <price>2.00</price>
  </part>
</order>
20 <order id="_5449432">
    <datetime>Feb 1 2000 5:37PM</datetime>
    <notes>Take your time, this customer still hasn't paid last
invoice.</notes>
    <salesperson id="41">
25    <name>John Sparky</name>
    <department>Sales</department>
    <phone>90991235</phone>
  </salesperson>
  <customer id="909989">
30    <name>Kens Hardware</name>
    <address>99 Ken St, Sydney, 2000</address>
    <phone>90999101</phone>
  </customer>
  <part id="10969">
35    <name>Widget Rubber Seal</name>
    <quantity>200</quantity>

```

```

        <price>0.25</price>
    </part>
    <part id="10899">
        <name>Widget Spring</name>
5        <quantity>10</quantity>
        <price>4.00</price>
    </part>
</order>
</orderlist>

```

10

The Upload Servlet 3 uploads the specified XML file and calls a Parser Servlet 4 which reads the file and deciphers it to produce a Document Object Model (as defined by W3C). The Document Object Model (DOM) provides programmatic access to the structure and content of the data being imported. In practice this means converting it into a series of values representing each node of an information (DOM) tree; as shown in Fig. 2.

The values are then passed to an XMLToData Converter Servlet 5 which ensures the values retrieved from the Parser 4 are in the correct format to pass to the information tree. The tree may then be viewed by the user using a Display Tree Servlet 6.

If the tree is to be saved it is written to temporary storage 7. The temporary storage areas basically consist of four tables ie ww_form-temp (metadata). ww_form_item_temp (metadata). ww_files_temp (data) and ww_objects_temp (data); the table structure is shown in Fig. 3.

Upon saving the XML tree the metadata and data values need to be stored. The relationship between parent-child and individual fields on a form is quite simple. All tags that appear at the same tree level are fields on the same form. If a tag is identified then it has a parent node.

Once an XML document has been received from an external source it can be fed into a data driven application comprised of:

- o Metadata – The forms (templates) required to publish content
- o A Home – The folders defined to hold the published content
- o Search Facilities – Automatic access to search facilities specifically tailored for the structure of the content published.
- o Content – The published content.
- o Workflow – A workflow process to direct published content.

This task involves the following steps:

1. Create new metadata (Form templates) by analysing the DOM's structure.

Given that XML data is hierarchical in structure, the metadata produced
 5 will also be hierarchical, that is, the forms will be built on parent/child
 relationships. Identifying the forms required involves a traversal of the DOM
 tree using the following criteria:

Start with the root node.

Any node with only a single value becomes a new field on the
 10 current form.

Any node with more than one child (the value of a node is
 represented as a child) requires a new form, a child form.

This process is recursive as we walk through the DOM structure:

```

15         begin
            node = getRootNode
            createForm(node)
        end

20         sub createForm(node)
            begin
                for each child of this node
                    if child node has more than one child of it's own
                        newForm = createForm(child)
25                 thisForm.addChild(newForm)
                    else
                        newField = createField(child)
                        thisForm.addField(newField)
                    endif
30             endfor
        end
    end
  
```

Given this process and the sample XML document presented, the forms
 shown in Figure. 4 can be identified:

35

2. Create a home for it and associated workflow.


```

        newDocField = createDocField(childnode, formfield)
        thisDocument.addField(newDocField)
    endfor
endfor
5   for each child form of the current form
    get all children of current node that have same name as
    child form for each child node
        newDocument = createDocument(childnode,
        childform)
10    thisDocument.addChild(newDocument)
    endfor
endfor
end

```

15 Given this process and our sample XML document, the documents shown in Figure. 5 would be produced.

 Having created the building blocks it remains to map the objects created to the underlying relational database.

20 It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

Dated this tenth day of March 2000

i - Launch Pty Ltd
Patent Attorneys for the Applicant:

F B RICE & CO

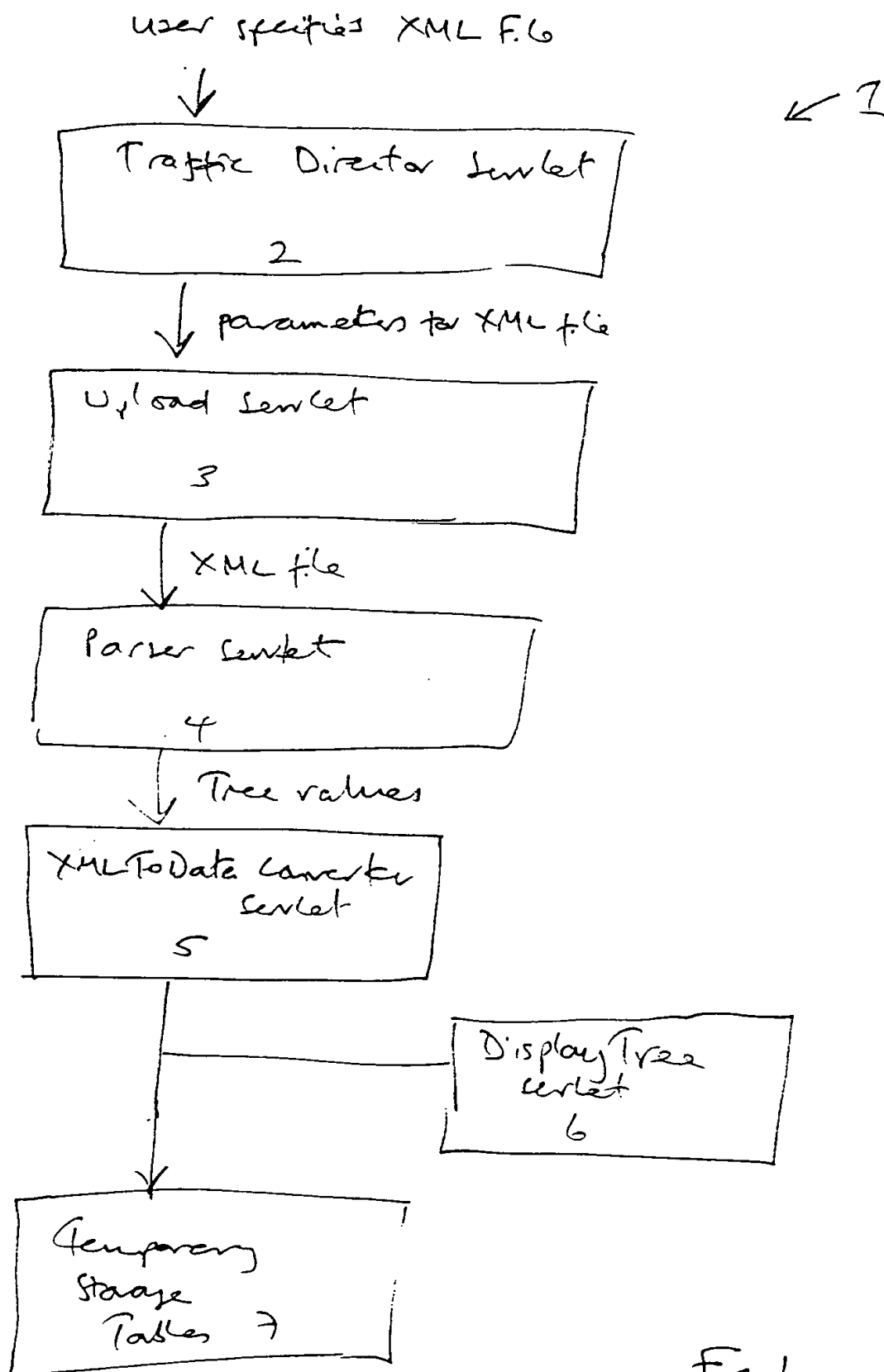


Fig. 1

Parameters In	Type Of Field
XMLFileName	Clob (stream of characters)

Parameters Out	Type Of Field
XMLNodeId	Integer (long)
XMLParentNodeId	Integer (long)
XMLNodeName	String (2000)
XMLUrlToPerform	String (2000)

Fig. 2

Table Name - ww_form_temp	
Column Name	Type
form temp id	Number(9)
User id	Varchar2(50)
Form name	Varchar2(255)
Description	Varchar2(512)
Title	Varchar2(2000)
Title XML tag	Varchar2(255)
Parent form temp id	Number(9)

Table Name - ww_form_item_temp	
Column Name	Type
form temp item id	Number(9)
Form temp id	Number(9)
Item text	Varchar2(255)
Item type	Varchar2(1)
Item XML tag	Varchar2(255)

Table Name - ww_files_temp	
Column Name	Type
form temp id	Number(9)
Folder id	Number(9)
Form temp id	Number(9)
title	Varchar2(2000)

Table Name - ww_objects_temp	
Column Name	Type
Object temp id	Number(9)
File temp id	Number(9)
form temp id	Number(9)
value	Varchar2(2000)

Fig. 3

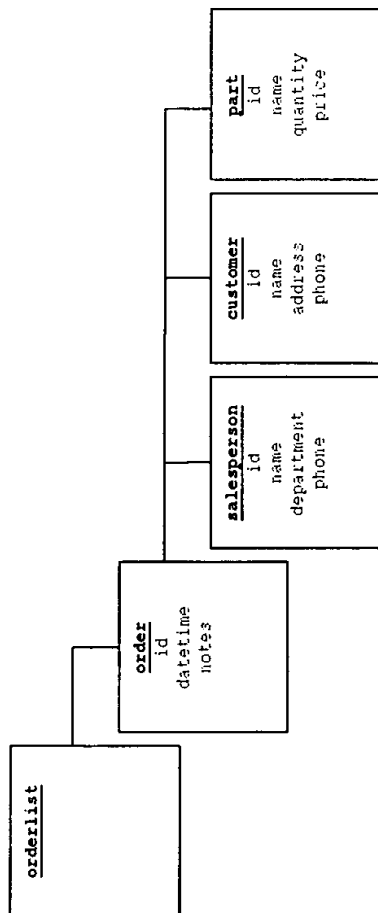


Figure 4

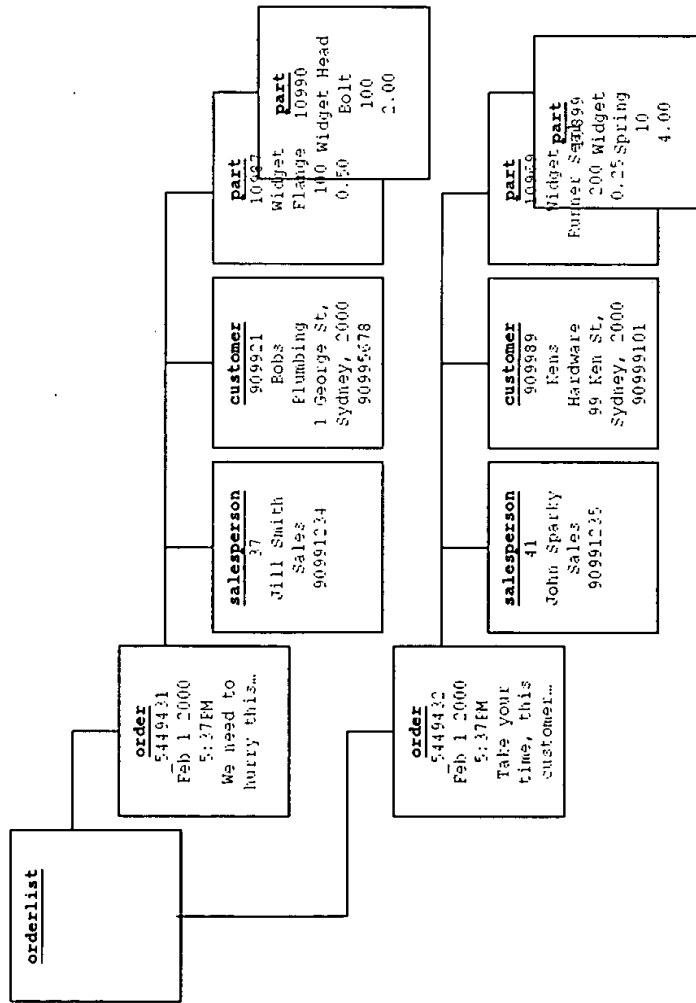


Figure 5